An extensible Set-Top-Box Architecture for interactive and broadcast Services offering sophisticated User Guidance

Frank Lonczewski, Rudolf Jaeger
BetaResearch
Betastrasse 1
85774 Unterfoehring, Germany

ABSTRACT

Currently available Set-Top-Boxes (STBs) are mainly used for digital TV reception. The User Interface (UI) and the UI dialog of such a device usually focus on its technological aspects and to a large degree ignore the needs of the user. The impact is that the user quite often is unsatisfied when interacting with the device. Recent UI design approaches [9][10] are proving that the focus should be put on the tasks that the user can perform with a STB rather than its underlying technical capabilities. However, modern design approaches both for the STB UI as well as its underlying system architecture can be incorporated into a complete system design with reasonable effort. This paper presents the architecture of the "d-box", a STB being used for broadcasting TV services as well as future interactive services like internet, home-shopping etc. in Germany. The layered software architecture employs a Java Virtual Machine offering a high degree of independency of its underlying hardware. Emphasis is put upon the user-friendliness of the device by providing a uniform UI both for experts and novices, therefore allowing intuitive and easy access for different user groups.

1. INTRODUCTION

Set-Top Boxes (STBs) employed in Digital Video Broadcast (DVB) Networks in the past were mainly used for digital TV distribution. They were simply designed to receive and decode digital encoded video-/audio-data streams [5]. Advances in the network standardisation process [4] supporting broadcast and IP-transparent data transmission and the availability of return channel capability offer new interactive services. Currently a significant growth in applications where data transfer is involved in conjunction with video/audio distribution is appearing. A simple example of such applications is internet/WWW access while watching TV. The advancements of the network infrastructure and the introduction of new services [7] require modifications of the software architecture of currently available STBs. Their software architecture and implementation shows a strong monolith structure [11]. In particular there is no clear separation between the hardware related software modules and the rest of the software. In consequence, the support of a new hardware platform requires a lot of effort for adapting pre-existing software. Therefore it is desirable to have a platform independent software architecture with a clear separation between the Application-, Application Service- and the Infrastructure Service-Layer. With the introduction of the Java™ language [1] it is possible to reach this goal. In order to fulfil both requirements, supporting new services and platform independency, the architecture of modern STBs should follow three fundamental design issues:

- A clear separation between the Infrastructure Service Layer (ISL), the Application Service Layer (ASL) and the Application Layer (AL) on top of the latter must be ensured.
- The ASL should provide a generic interface that platform independent applications can be built upon.
- The UI should be user task driven rather than technology-driven.

Figure 1 shows the principal architecture of modern STBs taking into account the above mentioned design rules. The ISL provides the necessary means being used by the ASL which in turn offers services accessible by all applications. Furthermore a clear separation of the ISL from the ASL also supports the roles of future business models which also differentiate between Infrastructure Service Provider and Application Service Provider.

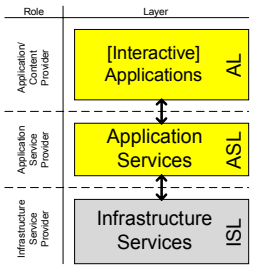

Figure 1. Layered architecture of modern STBs

The architecture of the presented STB called "d-box" is sold on the German market for free- and pay-TV. It clearly follows the above mentioned criteria thus offering the necessary means for the development of user friendly applications which can be run on different underlying hardware and operating system platforms. The following chapter presents an overview of the d-box architecture and describes the functionality of the various subsystems. In section 2.3 an in-depth description of the modules within the ASL is provided. In section 2.4 emphasis is put upon the Application Layer which incorporates some degree of user support for different user populations. Chapter 3 summarises the characteristics of the d-box and gives some indication of enhancements planned for the future.

2. STB ARCHITECTURE

Driven by the above mentioned requirements for the system architecture, the STB architecture can be divided into four layers, namely the Hardware Layer (HL), the Subsystem Service Layer (SSL), the Application Service Layer (ASL) and the Application Layer (AL), cf. figure 2.

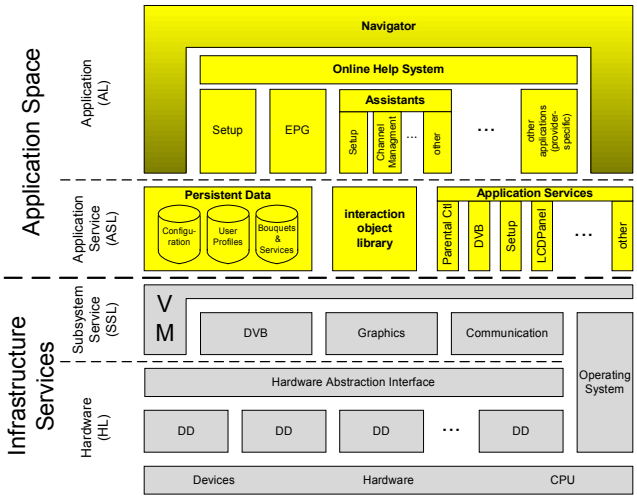

Figure 2. Layers of the d-box architecture

2.1 Hardware Layer (HL)

All hardware-dependent components are organized in the HL, namely the CPU, available I/O devices (e.g. the LCD panel) together with their Device Drivers (DD) as well as the Operating System (OS) of the STB. A clear separation of all hardware related functions is accomplished by the Hardware Abstraction Interface. It provides a generic interface to all necessary DDs (Video/Audio, Front-End, Ethernet, etc.) and to the Subsystem Service Layer (SSL). Thus a change of the underlying hardware only affects the associated DDs, no changes within higher layers are necessary. In conjunction with the OS the API calls of the Hardware Abstraction Interface work in a multithreaded fashion, thereby allowing access to specific DDs by several subsystems at the same time.

2.2 Subsystem Service Layer (SSL)

The SSL basically is composed of four modules and forms a common basis for the ASL.

DVB Subsystem

The DVB subsystem is responsible for filtering and processing all DVB specific data [2][3]. These are the so called Program Specific Information (PSI) and Service Information (SI). They are being used for the configuration of the STB as well as for service selection and the display of channel content information used by the Electronic Program Guide (EPG).

Communication Subsystem

The Communication Subsystem uses a TCP/IP protocol stack as a common interface to upper layers. Three physical transport mechanism are currently being implemented. The standard modem module (V.90) allows the interface to the PSTN network. In order to enable IP transparent information transfer over satellite an IP-decapsulation module [4] was implemented. Thus a high speed downlink to the STB is possible using the PSTN channel both for data requests and acknowledgements. The third module supports the data carousel mechanism [6] which is being used for:

- downloading software modules (including system software updates),
- downloading various application modules written in Java or simply for
- data broadcasting services (e.g. for a newsticker application, etc.).

Graphic Subsystem

The Graphics Subsystem implements a 2D graphics engine which consists of a window server system and a graphic library containing all necessary graphic primitives. A framebuffer DD interfaces with the underlying graphics hardware. The Java drawing primitives are mapped to the window server API to access the corresponding drawing primitives of the graphics library.

Virtual Machine (VM)

All three modules discussed above are interfaced to the Java VM. The Java applications running on the STB are utilizing the VM. Therefore all platform functionality used by Java applications is encapsulated by the VM which in turn uses some libraries contained both in the SSL and the ASL.

2.3 Application Service Layer (ASL)

The ASL components depicted in figure 2 are an Interaction Object Library, a number of Persistent Data Stores as well as a number of Application Services which can be used by applications built upon the ASL.

Interaction Object Library

The ASL contains a library of reusable interaction objects (“widgets“) which are used to build the UI of individual applications. The interaction object library is modeled according to the MVC paradigm [8]. In comparison to other available widget sets (e.g. Swing [12]), it is not possible to change the look & feel of individual interactive applications during runtime. Besides the fact that leaving out the support for re-evaluating the layout at runtime minimizes the feedback time of the STB in response to user interactions, some user studies currently on their way support this implementation decision since they seem to prove that typical users in the digital TV application domain have very little use of such a feature. However, from the standpoint of different providers writing applications for the STB, the look & feel of the interaction objects used should be handled flexibly since otherwise this would leave very little freedom for individual providers to distinguish from one another. Therefore, different look & feels are supported, but after instantiation of a “specific“ look and feel no support is available to change it on-the-fly.

Persistent Data Stores

Several persistent data stores are maintained in the ASL. In the context of this paper, we describe the persistent data stores required to access digital TV programs as well as those that store information about individual user settings.

The leftmost data store inside the ASL depicted in figure 2 contains the various configuration settings for the STB's individual hard- and software-components (e.g. configuration settings for connected antennas, modem settings, ethernet settings, etc.). Besides these, it also has to maintain configuration settings for other devices in the environment of the STB which can be controlled by it (e.g. a TV set, a videorecorder and/or other household appliances furnished to be controlled by the STB). The rightmost data store contains physical parameter sets required to tune the device to different broadcaster channels, together with other information that can be used by an EPG (e.g. bouquets[1], channel numbers, visibility attributes) to offer easy and comfortable access to digital TV content for the user. The remaining data store holds data values that try to approximate various attributes of individual users (e.g. attributes that describe the user's preferences, in part configured by the individual user herself as well as attributes that are evaluated by observing the user's interactions). By exploiting the collected data, the STB can offer sophisticated user guidance (e.g. intelligent help [10] and assistant-based user access) under user control.

Application Services

The STB facilitates a strong resource and application life cycle management. In order to fulfill this goal, all platform functionality is encapsulated in services which are controlled by a central interactive application called "Navigator". The Navigator controls the access and resource privileges for all other applications. The one and only way for individual applications to gain access to platform functionality is to instantiate service clients controlled by the Navigator.

The following application services are a representative set of all services available in the ASL:

- *Parental Control*: According to DVB, it is possible to signal a parental control age with a broadcasted event. This service encapsulates the functionality that applications can use to query the state of the parental control facilities of the STB.
- *DVB*: This service provides access to all relevant DVB data structures for individual broadcasting channels as well as access to detailed information for individual events. Furthermore, tuning and channel acquiring are also handled through this service.
- *Setup*: Applications are allowed to query the configuration settings of the STB. In addition, individual applications might also have the required access privileges to change some of the settings maintained in the configuration data store.

[1] In the context of DVB, a bouquet is a meta data structure bundling together several services, possibly offered by different content providers.

- *LCDPanel*: The LCD panel can be regarded as a separate output channel. Individual applications can gain access to the panel to display text and graphics by using this service.

2.4 Application Layer (AL)

The Navigator controls the life cycle of all other applications running on the STB. It also manages the resources of the STB by observing their usage in the AL. Since Application Services provide the only way to access system resources by interactive applications, the Navigator can exercise a very fine grained control over the application space of the STB. If a resource conflict occurs and cannot be resolved, the Navigator even has the possibility to terminate an application.

Together with the Navigator a number of resident applications provide user access to main STB functionalities. The most important applications of these are the Setup and the EPG. The former provides the functionality to configure various aspects of STB hard- and software modules stored in the respective data stores, whereas the latter allows the user to gain access to digital TV services like NearVideo on Demand (NVoD) and/or multifeed channels belonging to one TV service.

In comparison to various other approaches, the STB architecture under discussion provides more than one way for the user to access its features, thereby directly reflecting that both "novices" as well as "experts" are every-day users of the STB as an every-day device. To offer appropriate support for the novice, a number of specialized "assistant" applications (e.g. for configuring the device, for managing the channel list of the device, etc.) are available. Since the primary directive of all the assistants is to offer easy and guided access to the STB, only a subset of relevant features can be accessed, employing a very literate UI dialog-style. In contrast to this, the expert UI dialog allows a broader and more direct access to STB features, employing a more terse dialog-style well suited for expert users. An example is shown in figures 3 and 4.

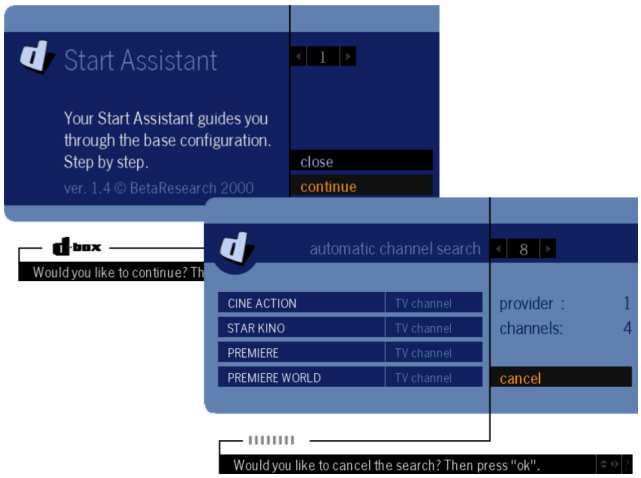

Figure 3. Assistant UI dialog excerpt

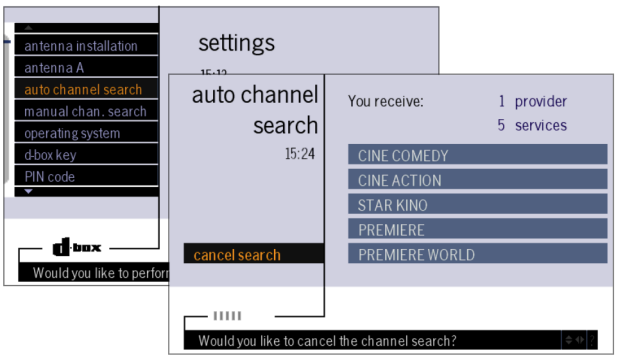

Figure 4. Expert UI dialog excerpt

The figures show two ways to execute the user task "perform automatic channel search". In the "Start Assistant" (cf. figure 3), the user cannot modify parameters that are used for the channel search (e.g. number of antennas connected, installation options, etc.). Rather, reasonable defaults for the parameters stored in the configuration data store are used for the operation. The user is fully guided through the whole configuration task. All he usually has to do (e.g. in the absence of any error situation) is to repeatedly press the "Ok" button on the remote control. In contrast to this, the expert settings offer direct access to all user configurable parameter settings of the STB. In the UI dialog no user guidance is employed. Figure 4 shows a menu with some of the user configurable STB settings as well as an expert UI dialog excerpt for the automatic channel search user task.

Further user support is given through an integrated on-line help system spanning across the whole interactive application space. The on-line help system is dynamic in its nature since other, temporarily downloaded applications can integrate their on-line help easily into the existing system.

3. SUMMARY

The major advantage of the modular and layer-oriented architecture of the STB presented is that it is highly independent of any specific hardware and OS platform. Thus the hardware development cycles which are a magnitude shorter than the software development cycles are fully utilized by this approach. Furthermore the architecture supports the differentiation between ASPs and ISPs (cf. figure 1).

The user experience we have studied so far is very promising with respect to the decision to both support expert and novice access to the STB. However, larger studies have to be performed to find out where the UI and the UI dialog can be further improved to match the expectations of different user groups. The material collected so far indicates that the user expects a more intelligent behaviour of the device in supporting typical tasks (e.g. reordering the channel list with respect to user preferences, accessing TV content with respect to individual user preferences, etc.) The assistant framework and the on-line help system as STB applications currently under development seem to be the natural places to support a broad range of intelligent behaviour. With the integration of modern UI design approaches like CADUI (Computer Aided Design of User Interfaces) [9] together with intelligent user guidance [10] and on-line help as well as the advancements in communication networks, the future of STBs in the multimedial, digital TV domain seems to be clear: they will subsume traditional, but previously separated communication means (e.g. telephone, newspaper, magazine, internet services, etc.) under one homogeneous roof.

4. ACKNOWLEDGEMENTS

The STB presented herein summarizes the work of many people. We specifically would like to thank Michael Brandt, Martin Mohring, Wolfram Proske, Hartmut Wolf and Jürgen Zeller for their major contributions to the project as well as the whole d-box development team for their effort in realizing it.

5. REFERENCES

[1] Arnold K. and Gosling J. *The Java™ Programming Language, Second Edition*. Addison-Wesley, Reading, Massachusetts, 1998

[2] ETR 211. *Guidelines on Implementation and Usage of Service Information (SI)*. August 1997

[3] ETR 300 468 *Specification for Service Information in DVB Systems*. January 1997

[4] ETSI EN 301 197. *DVB Specification for Data Broadcasting*. (Draft), June 1997

[5] ISO/IEC 13818-1. *Information technology - Generic coding of moving pictures and associated audio – System*. JTC1/SC29/WG11, November 1994

[6] ISO/IEC 13818-6 *Information technology - Generic coding of moving pictures and associated audio information - Extension for Digital Storage Media Command and Control*. November 1995

[7] Jaeger R. and Neubauer J. *Broadband Internet Access in a DVB Network*. to appear in ISSLS 2000 (Stockholm, Sweden, 18-23 June 2000)

[8] Krasner G.E. and Pope T.P. *A cookbook for using the model-view controller user interface paradigm in Smalltalk-80*. Journal of Object-Oriented Programming, August/September 1988, 1(3), pp. 26-49

[9] Lonczewski F. and Schreiber S. *The FUSE-System: an Integrated User Interface Design Environment*. in Proceedings of the 2nd International Workshop on Computer-Aided Design of User Interfaces CADUI'96 (Namur, 5-7 June 1996), J. Vanderdonckt (Ed.), Presses Universitaires de Namur, Namur, 1996, pp. 37-56

[10] Lonczewski F. *Providing User Support for Interactive Applications with FUSE*. in Proceedings of Intelligent User Interfaces IUI'97 (Orlando, 6-9 January 1997), J. Moore, E. Edmonds, A. Puerta (Eds.), ACM Press, New York, USA, 1997, pp. 253-256

[11] Pekowsky S. and Jaeger R. *The Set-Top-Box as a Multi-Media-Terminal*. IEEE Transactions on Consumer Electronics, August 1998, Vol. 44, Nr. 3, pp. 833-840

[12] Walrath K. and Campione M. *The JFC Swing Tutorial – A Guide to Constructing GUIs*. The Java™ Series, Addison-Wesley, Reading Massachusetts, 1999